Supplementary Information for

# Physics filtering favors the generalization of robot learning

Jindou Jia *et al.*

**This PDF file includes:**

Supplementary Methods Sections A-G

Supplementary Figures S1 to S4

Supplementary Table S1

Supplementary References

**Other supplementary material for this manuscript includes the following:**

Supplementary Movies 1 to 4

Supplementary Movie 1: Quadruped experiments

Supplementary Movie 2: Quadruped simulations

Supplementary Movie 3: Drone maneuvering experiments

Supplementary Movie 4: Aerial manipulator experiments

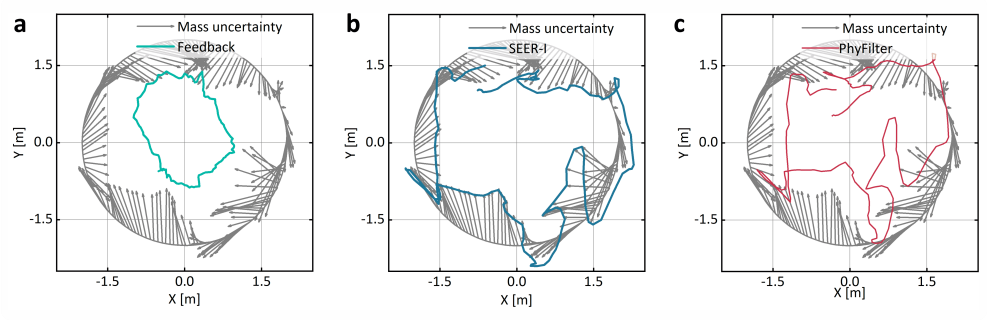


**Fig. S1 Estimation performance of the drone example.** **a**, Uncertainty estimation with feedback-based disturbance observer. **b**, Uncertainty estimation with SEER-I. **c**, Uncertainty estimation with the proposed PhyFilter. For reference, the ground truth of mass uncertainty is provided in (**a**)-(**c**), which can be deduced from the ground-truth mass in advance. It can be seen that SEER-I can only reflect the mass uncertainty, while PhyFilter can further capture other unknown wind and inherent uncertainties, leading to more accurate trajectory tracking, as shown in Fig. 4e.

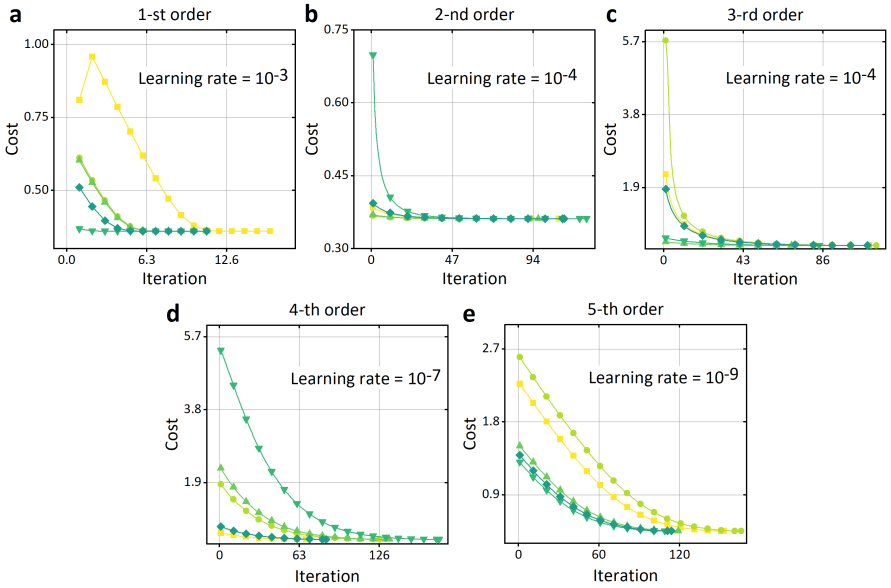


**Fig. S2 The convergence performance of Algorithm 2 at different filter orders in the acceleration example.** Each case is repeated 5 times with different initial filter parameter $\boldsymbol{a}_{f0}$. A higher filter order requires a smaller learning rate.

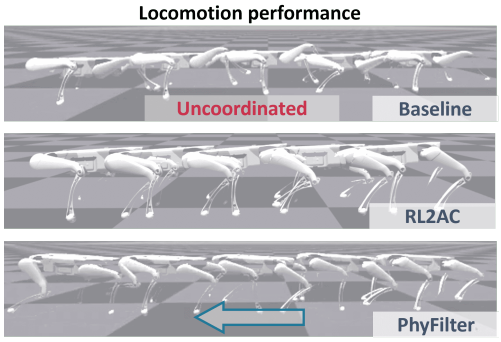


**Fig. S3 The trained locomotion performance of all compared methods.** PhyFilter and RL2AC can yield more coordinated locomotions, whereas the baseline results in inferior performance.

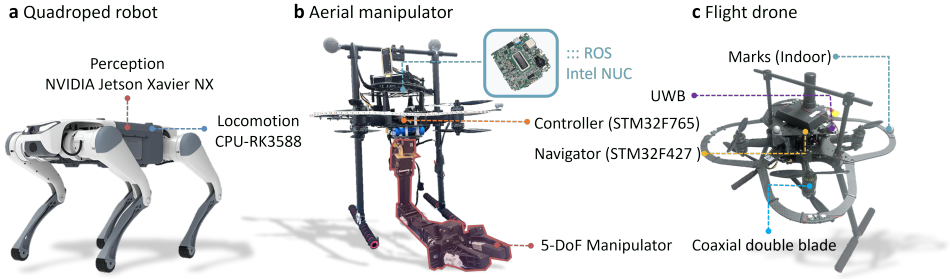


**Fig. S4 The employed hardwares.** **a**, The quadruped robot used in the locomotion example. **b**, The aerial manipulator used in the manipulation example. **c**, The flight drone used in the flight example.

| **Training parameter** | **Value** | **Domain randomization** | **Range** |
|---|---|---|---|
| Number of environments | 4096 | Mass | $-1 \sim 3$ kg |
| Number of iterations | 15000 | Velocity | $0 \sim 0.8$ m/s |
| Filter order $n$ | 1 | Gain $\boldsymbol{K}_p$ | $0.8 \sim 1.2$ |
| Filter parameter $a_0$ | 1 | Gain $\boldsymbol{K}_d$ | $0.8 \sim 1.2$ |
| | | Friction $\boldsymbol{K}_d$ | $0.1 \sim 1.25$ |

**Table S1** Hardware of test drones.

# Section A Related work

## A.1 Policy learning for quadruped robots

Benefiting from cheaper yet more efficient hardware (e.g., proprioceptive actuators [1]), advanced control algorithms around optimal control and RL, and high-fidelity simulators (e.g., MuJoCo [2], Isaac Gym [3]), quadruped robots are currently seeing resurgent interest [4]. In particular, RL is currently the most mainstream algorithm with various implementation manners, like imitation learning [5], hierarchical learning [5, 6], privileged learning [7, 8], and competitive learning [9]. Despite the fact that adding noise during training can improve the robustness, there is still a major sim-to-real gap in scenarios that exceed the training distribution/training scenarios [10].

Nowadays, plenty of researchers attempt to incorporate classical control schemes and other *a priori* physical knowledge into RL to improve the learning generalization [11]. For example, [12] discovers that the contact force of the quadruped robot under an RL-based PD controller can be approximated by a linearly parameterized form with respect to the joint tracking error. Hence, a composite adaptive controller can be naturally employed to estimate the unseen disturbances, narrowing the sim-to-real gap. In [13], a bio-inspired gait scheduler is incorporated into RL to encode pseudo gait procedural memory and adaptive motion adjustment, leading to an improved generalized performance for quadruped locomotion. [14] employs a model-based planner to guide the learning policy, so that the sparse reward problem of RL can be avoided while maintaining robustness. Our presented PhyFilter also attempts to incorporate a control feedback scheme into RL, yielding a more lightweight and interpretable framework.

## A.2 Dynamics learning for multirotor drones

Achieving precise maneuver control of drones requires effective handling of external disturbances and internal model uncertainties. This part mainly reviews the learning-dominant methods. [15] designs an end-to-end control strategy for the quadrotor by employing RL, which can achieve the level of human world champions in drone competitions. To address uncertainties, this work trains the control strategy using the domain randomization [16] and constructs an empirical noise model to reduce the difference between simulation and reality. [17] employs residual networks to estimate dynamical uncertainties explicitly, which is then regarded as the learning input of an end-to-end RL. [18] compares the performance of RL and optimal control in drone competitions. This work finds that RL has a higher success rate and maneuvers faster in the presence of uncertainties. A neural network-based estimation algorithm is proposed in [19] to predict the downwash-induced disturbance forces and torques acting on vertically stacked drones. The results demonstrate that incorporating *Gaussian*-predicted airflow velocities as input features significantly enhances the estimation accuracy.

The approach of integrating traditional control and advanced learning for the multirotor drone control is increasingly appealing [20]. $\mathcal{L}_1$ adaptive observer-enhanced RL is presented in [21], enabling the drone to precisely control under multiple disturbances.

Whereas, the estimation error of $\mathcal{L}_1$ adaptive observer is not considered in RL training. The *Koopman* operator is employed in [22] to learn the uncertainty model online, favoring the convergence of a model-based uncertainty observer. It can be shown that the *Koopman* operator-based observer constitutes a special case of PhyFilter.

## A.3 Precise control of aerial manipulators

Different from bare multirotors, achieving precise control of an aerial manipulator presents substantial challenges [23–25]. Firstly, the control accuracy of the aerial platform is highly susceptible to model uncertainties. In practice, the assumption that a three-dimensional model cannot perfectly match reality due to the inevitable presence of motors, sensors, and other components [26–28]. In addition, the dynamics of the aerial platform and the manipulator are intrinsically coupled, as the multi-link manipulator is rigidly mounted on the drone. As pointed out in [29, 30], the strong dynamic coupling disturbances are significantly influenced by the relative motion of the manipulator with respect to the aerial platform. The strong coupling effects further exacerbate the instability of the aerial platform. The reason is that explosive dynamic coupling terms affect its dynamics. Beyond this, external disturbances like aerodynamic effects play a critical role in degrading the dynamic performance of the aerial platform [30–33].

A common approach to address coupling disturbances is the use of disturbance observers. For instance, an adaptive sliding-mode disturbance observer with a prescribed convergence-time bound is proposed in [34] to estimate state-dependent uncertainties and external disturbances. In [33], an acceleration-based estimation method is presented to estimate total external wrench including modeling errors and wind disturbances. However, when external disturbances are coupled with system states, the assumption of bounded uncertainty derivatives becomes impractical. This is primarily because one cannot assume that the states are bounded before the closed-loop stability of the system is established [35, 36]. In [37], a characteristic modeling with the bounded identification error is introduced into the integral controller to guarantee the finite-time stability of the aerial manipulator. Furthermore, in [38], two extended state observers are incorporated into a cascade control scheme to improve quadrotor performance by accounting for first-order coupling effects. Nevertheless, these ESOs rely on a uniform regulating mechanism to adjust different observed states. Such a framework may lead to inappropriate parameter tuning, thereby degrading the dynamic performance of the observers. Although observer-based estimation techniques have yielded promising results, the prior knowledge of the system dynamics is not fully exploited to enhance estimation accuracy [33–36, 38].

Driven by the availability of large-scale datasets and advances in modern learning algorithms, data-driven learning has made remarkable progress. Preliminary efforts have been devoted to disturbance learning. For example, a radial basis function neural network with an adaptive weight update law has been developed to compensate for lumped disturbances, including model uncertainties and coupling effects, in real time [27, 39]. In [40], an adaptive approach is presented to compensate for the unmodeled nonlinear disturbances brought by the manipulator motion. Moreover, a gaussian process regression is developed to estimate the external environment disturbances [32].

The well-known generalization problem arises when the system encounters scenarios beyond the training dataset [41, 42]. Although offline training algorithms can be used to initialize neuron weights [27, 43], the inevitable online adaptation required to cope with time-varying disturbances often results in sub-optimal transient responses.

## A.4 Online differentiator

Calculating the precise derivative of a measured signal in real time is a crucial topic with numerous practical uses, including reliable state estimation. The most straightforward analytical differentiator is the finite difference-based discrete differentiator (DD), which approximates the derivative of discrete-time signals using differences between their sampled values. As a core component of active disturbance rejection control, the tracking differentiator (TD) [44] performs differentiation on noisy signals through a specially designed *sign* function. [45] propose a robust differentiator (RD) capable of converging to the derivative of a given signal within a predefined time frame. Nevertheless, these aforementioned analytical algorithms involve cumbersome parameter tuning procedures, and variations in signal characteristics may necessitate corresponding parameter adjustments. Learning-based methodologies represent another effective approach for deriving the derivative of measured signals. For instance, [46] formulates a neural acceleration estimator to achieve accurate prediction of the motion of uneven free-flying objects. However, as with other neural network-based application scenarios, the generalization problem is an intractable difficulty that cannot be avoided.

## A.5 Physics-informed machine learning

Physics-informed machine learning [16, 41, 47–53] is designed to integrate prior physical insights into the development of machine learning methods. The primary objectives of this integration include enhancing model interpretability, accelerating training convergence, boosting predictive precision, and strengthening generalization capabilities. Three synergistic strategies are commonly adopted. These involve introducing observational [16, 48], inductive [41, 49], and learning [50–53] biases into the training dataset, learning structure, and optimization algorithm, respectively. For instance, [54] demonstrates that neural networks can be trained to directly approximate the mass matrix and potential energy functions derived from *Lagrangian* and *Hamiltonian* mechanics. Such physics-informed structures yield more physically consistent models compared to conventional black-box neural network architectures. Along a parallel research trajectory focused on neural ODEs [49], [52] propose the *Hamiltonian* neural network, which is specifically engineered to learn the *Hamiltonian* of dynamic systems. A key advantage of this approach is its ability to preserve the energy conservation property inherent to such systems. However, a notable constraint of the *Hamiltonian* neural network is its reliance on canonical coordinates. To address this limitation, [53] develops the *Lagrangian* neural network, which eliminates the need for canonical coordinates.

In the domain of learning-based robotic autonomy, where improving generalization remains a critical challenge, [51] introduces a self-supervised neural-symbolic learning framework termed imperative learning. This framework formulates the learning task

as a bilevel optimization problem, wherein the base learner leverages symbolic knowledge, encompassing physical principles and logical reasoning, to enhance performance. [55] explores the integration of model predictive control (MPC) with actor-critic RL. Their findings indicate that this hybrid approach enables the simultaneous retention of MPC's short-term optimization strengths and the end-to-end training benefits characteristic of RL. A physics-informed world model is developed in [56] for bib-prehensile manipulation, in which a physics-aware randomization strategy is employed to bridge the sim-to-real gap. Inspired by a physics-based aerodynamic model, [19] incorporates airflow velocities as feature inputs to the neural network, enabling precise estimation of disturbance force and torque for vertically stacked drones.

## A.6 Generalization improvement

### A.6.1 Domain randomization

To enable data scaling, various simulational physics engines [2, 3, 57] have been established to train robotic agents. The utilized data is synthesized from massively different settings by randomizing the parameters of the simulator, also known as domain randomization [16, 48], aiming to bridge the sim-to-real gap. Several drawbacks exist in the paradigm. At first, domain randomization pursues the average performance among randomized environments, i.e., sacrificing accuracy for robustness [4, 41, 58], similar to classical robust control [4, 58]. Secondly, substantial computing resources are often required to train such a monolithic model, which may be infeasible for compute-constrained cases. Finally, universal physics engines are incapable of encompassing the full spectrum of diverse real physical robots and their dynamical environments. The standardization of data format, software, and hardware still remains a long-term endeavor [59].

### A.6.2 Domain adaptation

Apart from domain randomization, domain adaptation [60] is another way to combat the generalization problem, which tries to identify the encountered environment explicitly, so that current decisions can be appropriately adjusted [60]. This scheme is akin to adaptive control [4, 61]. In comparison with domain randomization, while domain adaptation is trained more complexly, it maintains accuracy in specific scenarios. A particular online domain adaptation case, test-time adaptation [60], aims at updating the pre-trained model only through unlabeled test data, considering the source data is usually inaccessible during online testing [62]. However, domain adaptation also relies on substantial computing resources to train a monolithic model.

### A.6.3 Filtering learning output

Few previous works use physics knowledge to filter the outcome of machine learning. A differentiable *Kalman* filter augmented neural acceleration estimator is designed in [46, 63], in which the *Kalman* filer balances the measurement (i.e., learning output) and prediction from *a priori* propagation model. However, the targeted uncertainty of the above method is limited to random white noise.

## Section B Formalization of parameter learning

The filter parameter $\boldsymbol{a}_f = \{a_0, a_1, \cdots, a_{n-1}\}$ in Eq. (3) can be set according to the mature *pole-placement* theory. To avoid the domain-specific expertise required, an auto-learning algorithm is further developed to learn the filter parameter $\boldsymbol{a}_f$ from the aspect of an optimal control scheme.

Before the procedure, define $\bar{\boldsymbol{\xi}} = \left[\boldsymbol{\xi}^T, \dot{\boldsymbol{\xi}}^T, \cdots, \boldsymbol{\xi}^{(n-1)T}\right]^T \in \mathbb{R}^{n_x n}$, $\boldsymbol{\alpha}_i = \int^i \hat{\boldsymbol{f}}(t)(dt)^i \in \mathbb{R}^{n_x}$, $\bar{\boldsymbol{\alpha}} = \left[\boldsymbol{\alpha}_1^T, \boldsymbol{\alpha}_2^T, \cdots, \boldsymbol{\alpha}_{n-1}^T\right]^T \in \mathbb{R}^{n_x(n-1)}$, $\boldsymbol{\beta}_i = \int^i \boldsymbol{f}_{\boldsymbol{\theta}}(t)(dt)^i$, and $\bar{\boldsymbol{\beta}} = \left[\boldsymbol{\beta}_1^T, \boldsymbol{\beta}_2^T, \cdots, \boldsymbol{\beta}_{n-1}^T\right]^T \in \mathbb{R}^{n_x(n-1)}$. Rearrange Eq. (5), leading to

$$\hat{\boldsymbol{f}}(\bar{\boldsymbol{\xi}}, \bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{\beta}}, \boldsymbol{a}_f) = a_0\boldsymbol{\xi} + \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) + a_0 \int^{n-1} \boldsymbol{x}(dt)^{n-1} - \sum_{i=1}^{n-1} a_{n-i}\boldsymbol{\alpha}_i + \sum_{i=1}^{n-1} a_{n-i}\boldsymbol{\beta}_i \quad \text{(B1)}$$

where

$$\begin{aligned}
\dot{\bar{\boldsymbol{\xi}}} &= \boldsymbol{\phi}_{\bar{\boldsymbol{\xi}}}(\bar{\boldsymbol{\xi}}, \bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{\beta}}, \boldsymbol{a}_f) = \left[\dot{\boldsymbol{\xi}}^T, \ddot{\boldsymbol{\xi}}^T, \cdots, \boldsymbol{\xi}^{(n-1)T}, (-\boldsymbol{g} - \hat{\boldsymbol{f}})^T\right]^T \\
\dot{\bar{\boldsymbol{\alpha}}} &= \boldsymbol{\phi}_{\bar{\boldsymbol{\alpha}}}(\bar{\boldsymbol{\xi}}, \bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{\beta}}, \boldsymbol{a}_f) = \left[\hat{\boldsymbol{f}}^T, \boldsymbol{\alpha}_1^T, \cdots, \boldsymbol{\alpha}_{n-3}^T, \boldsymbol{\alpha}_{n-2}^T\right]^T \\
\dot{\bar{\boldsymbol{\beta}}} &= \boldsymbol{\phi}_{\bar{\boldsymbol{\beta}}}(\bar{\boldsymbol{\beta}}, \boldsymbol{a}_f) = \left[\boldsymbol{f}_{\boldsymbol{\theta}}^T, \boldsymbol{\beta}_1^T, \cdots, \boldsymbol{\beta}_{n-3}^T, \boldsymbol{\beta}_{n-2}^T\right]^T .
\end{aligned} \quad \text{(B2)}$$

Regard the above equations Eqs. (B1)-(B2) as an optimal control system with concatenated states $\{\bar{\boldsymbol{\xi}}, \bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{\beta}}\}$ and control input $\boldsymbol{a}_f$. Then, the optimization problem to find optimal $\boldsymbol{a}_f$ can be formalized as

$$\begin{aligned}
\min_{\boldsymbol{a}_f} J(\boldsymbol{a}_f) &= \min_{\boldsymbol{a}_f} \sum_{i=1}^{N-1} l_i(\boldsymbol{f}_i^*, \hat{\boldsymbol{f}}_i, \boldsymbol{a}_f) + l_N(\boldsymbol{f}_i^*, \hat{\boldsymbol{f}}_i) \\
s.t. \quad \bar{\boldsymbol{\xi}}_{i+1} &= \boldsymbol{\phi}_{\bar{\boldsymbol{\xi}}}^d(\bar{\boldsymbol{\xi}}_i, \bar{\boldsymbol{\alpha}}_i, \bar{\boldsymbol{\beta}}_i, \boldsymbol{a}_f), \quad i = 1, 2, \cdots, N-1 \\
\bar{\boldsymbol{\alpha}}_{i+1} &= \boldsymbol{\phi}_{\bar{\boldsymbol{\alpha}}}^d(\bar{\boldsymbol{\xi}}_i, \bar{\boldsymbol{\alpha}}_i, \bar{\boldsymbol{\beta}}_i, \boldsymbol{a}_f), \quad i = 1, 2, \cdots, N-1 \\
\bar{\boldsymbol{\beta}}_{i+1} &= \boldsymbol{\phi}_{\bar{\boldsymbol{\beta}}}^d(\bar{\boldsymbol{\beta}}_i, \boldsymbol{a}_f), \quad i = 1, 2, \cdots, N-1
\end{aligned}$$

where $N$ denotes the sample number, $l_i(\cdot) \in \mathbb{R}$ is defined to quantify the state differences between model rollout $\hat{\boldsymbol{f}}_i$ and labelled sample $\boldsymbol{f}_i^*$, $l_N(\cdot) \in \mathbb{R}$ represents the terminal cost, $\boldsymbol{\phi}_{\bar{\boldsymbol{\xi}}}^d(\cdot)$, $\boldsymbol{\phi}_{\bar{\boldsymbol{\alpha}}}^d(\cdot)$, and $\boldsymbol{\phi}_{\bar{\boldsymbol{\beta}}}^d(\cdot)$ refer to the discretized integration of $\boldsymbol{\phi}_{\bar{\boldsymbol{\xi}}}(\cdot)$, $\boldsymbol{\phi}_{\bar{\boldsymbol{\alpha}}}(\cdot)$, and $\boldsymbol{\phi}_{\bar{\boldsymbol{\beta}}}(\cdot)$, respectively. Note that $l_i(\cdot)$ is chosen as $(\boldsymbol{f}_i^* - \hat{\boldsymbol{f}}_i)^T(\boldsymbol{f}_i^* - \hat{\boldsymbol{f}}_i)$ in this work for $i = 1, 2, \cdots, N$. For the convenience of subsequent expressions, define $\boldsymbol{\varsigma} = \left[\bar{\boldsymbol{\xi}}^T, \bar{\boldsymbol{\alpha}}^T, \bar{\boldsymbol{\beta}}^T\right]^T \in \mathbb{R}^{n_x(3n-2)}$, one can achieve

$$\min_{\boldsymbol{a}_f} J(\boldsymbol{a}_f) = \min_{\boldsymbol{a}_f} \sum_{i=1}^{N-1} l_i(\boldsymbol{f}_i^*, \hat{\boldsymbol{f}}_i, \boldsymbol{a}_f) + l_N(\boldsymbol{f}_i^*, \hat{\boldsymbol{f}}_i)$$

$$s.t. \quad \boldsymbol{\varsigma}_{i+1} = \boldsymbol{\phi}^d_{\boldsymbol{\varsigma}}(\boldsymbol{\varsigma}_i, \boldsymbol{a}_f), \quad i = 1, 2, \cdots, N-1 \tag{B3}$$

with $\boldsymbol{\phi}^d_{\boldsymbol{\varsigma}}(\cdot) = \left[\boldsymbol{\phi}^d_{\tilde{\boldsymbol{\xi}}}(\cdot)^T, \boldsymbol{\phi}^d_{\tilde{\boldsymbol{\alpha}}}(\cdot)^T, \boldsymbol{\phi}^d_{\tilde{\boldsymbol{\beta}}}(\cdot)^T\right]^T$. $\boldsymbol{\phi}^d_{\boldsymbol{\varsigma}}(\boldsymbol{\varsigma}_i, \boldsymbol{a}_f)$ is further simplified as $\boldsymbol{\phi}^d_{\boldsymbol{\varsigma},i}$ for convenience.

Next, the *Lagrange* multiplier method is utilized to solve Eq. (B3), define *Lagrange* function

$$\mathcal{L} = J(\boldsymbol{a}_f) + \sum_{i=1}^{N-1} \boldsymbol{\lambda}^T_{\boldsymbol{\varsigma},i}(\boldsymbol{\phi}^d_{\boldsymbol{\varsigma},i} - \boldsymbol{\varsigma}_{i+1})$$

with *Lagrange* multiplier $\boldsymbol{\lambda}_{\boldsymbol{\varsigma},i} \in \mathbb{R}^{n_x(3n-2)}$ for $i = 1, 2, \cdots, N-1$. The first-order optimality conditions of the learning problem can be derived as

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\varsigma}} = \mathbf{0}, \quad \frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}_{\boldsymbol{\varsigma}}} = \mathbf{0}, \quad \frac{\partial \mathcal{L}}{\partial \boldsymbol{a}} = \mathbf{0}.$$

Thus, one can render

$$\boldsymbol{\lambda}_{\boldsymbol{\varsigma},i-1} = \nabla_{\boldsymbol{\varsigma}_i} l_i + \boldsymbol{\lambda}^T_{\boldsymbol{\varsigma},i} \nabla_{\boldsymbol{\varsigma}_i} \boldsymbol{\phi}^d_{\boldsymbol{\varsigma},i}, \ \boldsymbol{\lambda}_{\boldsymbol{\varsigma},N-1} = \nabla_{\boldsymbol{\varsigma}_N} l_N, \tag{B4}$$

$$\boldsymbol{\varsigma}_{i+1} = \boldsymbol{\phi}^d_{\boldsymbol{\varsigma},i}, \quad \boldsymbol{\varsigma}_1 = \boldsymbol{\varsigma}(0), \tag{B5}$$

$$\nabla_{\boldsymbol{a}_f} \mathcal{L} = \sum_{i=1}^{N-1} (\nabla_{\boldsymbol{a}_f} l_i + \boldsymbol{\lambda}^T_{\boldsymbol{\varsigma},i} \nabla_{\boldsymbol{a}_f} \boldsymbol{\phi}^d_{\boldsymbol{\varsigma},i}) = \mathbf{0}. \tag{B6}$$

The optimal parameter $\boldsymbol{a}_f$ that minimizes $\mathcal{L}$ can be obtained by solving the linear system above. However, directly solving Eqs. (B4)-(B6) becomes computationally expensive, particularly when $N$ and $n_x$ are large. To address this, starting from an initial guess $\boldsymbol{a}_{f0}$, we employ an iterative scheme to compute $\boldsymbol{a}_f$.

At first, the analytical gradient computation $\nabla_{\boldsymbol{a}_f} \mathcal{L}$ is driven by sequentially doing forward rollout Eq. (B5) and backward rollout Eq. (B4), where the latter one is also known as the term *adjoint solve* or *reverse-mode differentiation* [49]. The calculation process of $\nabla_{\boldsymbol{a}_f} \mathcal{L}$ is summarized in Algorithm 1.

**Algorithm 1** Analytic gradient computation

**Input:** Learning objective $l_i(\cdot), l_N(\cdot)$; model $\boldsymbol{\phi}^d_{\boldsymbol{\varsigma}}(\cdot)$; continuous trajectories $\{\boldsymbol{f}^*(t), \boldsymbol{x}(t), \boldsymbol{z}(t)\}$.
**Result:** Gradient $\nabla_{\boldsymbol{a}_f} \mathcal{L}$.
1: **Initialize:** Parameter $\boldsymbol{a}_{f0}$.
2: $\boldsymbol{\varsigma} \leftarrow$ Forward rollout of $\boldsymbol{\phi}^d_{\boldsymbol{\varsigma}}(\cdot)$ using Eq. (B5);
3: Compute $\nabla_{\boldsymbol{\varsigma}_i} l_i$, $\nabla_{\boldsymbol{\varsigma}_i} \boldsymbol{\phi}^d_{\boldsymbol{\varsigma},i}$, $\nabla_{\boldsymbol{\varsigma}_N} l_N$, $\nabla_{\boldsymbol{a}_f} l_i$, $\nabla_{\boldsymbol{a}_f} \boldsymbol{\phi}^d_{\boldsymbol{\varsigma},i}$;
4: $\boldsymbol{\lambda}_{\boldsymbol{\varsigma}} \leftarrow$ Reverse rollout using Eq. (B4);
5: $\nabla_{\boldsymbol{a}_f} \mathcal{L} \leftarrow$ Compute gradient using Eq. (B6).

With updated $\nabla_{\boldsymbol{a}_f}\mathcal{L}$, the parameter $\boldsymbol{a}_f$ is optimized according to the gradient descent algorithm, which is summarized in Algorithm 2. Note that the gradient computation is only supported for a single continuous state trajectory, with computational complexity scaling linearly with trajectory length. In practical applications, multiple long-horizon trajectory segments may be employed. In Algorithm 2, the mini-batching and stochastic optimization methods are employed to facilitate learning results. The learning step can be set using *Adam* or other stochastic gradient descent-related approaches. An early stopping mechanism is adopted as the convergence condition, which can markedly improve the training efficiency.

**Algorithm 2** Training filter parameters

**Input:** Objective $l_k(\cdot), l_N(\cdot)$; mini-batch size $s$; trajectories $\{\boldsymbol{f}^*(t), \boldsymbol{x}(t), \boldsymbol{z}(t)\}$.
**Result:** Parameter $\boldsymbol{a}_f$.
1: **Initialize:** Network parameter $\boldsymbol{\theta}$; filter parameter $\boldsymbol{a}_{f0}$; slice $\mathcal{D}^{tra}$ into $M$ segments $\{\mathcal{D}^{tra}_{j=1,\cdots,M}\}$ with $s$ length each.
2: **repeat**
3: **for** $\{\boldsymbol{f}^*_{1:s}, \boldsymbol{x}_{1:s}, \boldsymbol{z}_{1:s}\}$ in $\{\mathcal{D}^{tra}_{j=1,\cdots,M}\}$ **do**
4: Compute analytic gradient $\nabla_{\boldsymbol{a}_f}\mathcal{L}$ using Algorithm 1;
5: Compute learning step $\eta \leftarrow Optimizer(\boldsymbol{a}_f, \nabla_{\boldsymbol{a}_f}\mathcal{L})$;
6: $\boldsymbol{a}_f \leftarrow \boldsymbol{a}_f - \eta$.
7: **end for**
8: **until convergence**

The learning performance of Algorithm 2 is verified in the acceleration example, as shown in Fig. 6e. More experimental results are provided in Supplementary Fig. S2.

# Section C Filtering learning residual with more general form

In comparison with Eq. (3), a more general filter is considered in this part, i.e.,

$$\mathcal{L}\left[\mathcal{F}(\boldsymbol{\gamma}(t))\right] = \frac{b_{n-1}s^{n-1} + \cdots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0}\Upsilon(s) \tag{C7}$$

with more parameters $b_{n-1}, \cdots, b_1, b_0$. The above filter can cover the bandpass case. By substituting Eq. (C7) into Eq. (1) and performing the result in time-domain, it can be rendered that

$$\begin{aligned}
&\hat{\boldsymbol{f}}^{(n)}(t) + a_{n-1}\hat{\boldsymbol{f}}^{(n-1)}(t) + \cdots + a_1\hat{\boldsymbol{f}}'(t) + a_0\hat{\boldsymbol{f}}(t) \\
=&\boldsymbol{f}^{(n)}_{\boldsymbol{\theta}}(t) + a_{n-1}\boldsymbol{f}^{(n-1)}_{\boldsymbol{\theta}}(t) + \cdots + a_1\boldsymbol{f}'_{\boldsymbol{\theta}}(t) + a_0\boldsymbol{f}_{\boldsymbol{\theta}}(t) \\
&+ b_{n-1}\boldsymbol{\gamma}^{n-1}(t) + \cdots + b_1\boldsymbol{\gamma}'(t) + b_0\boldsymbol{\gamma}(t).
\end{aligned} \tag{C8}$$

Define $\boldsymbol{\chi}_1(t) = a_{n-1}\int \hat{\boldsymbol{f}}(t)dt + \cdots + a_1\int^{n-1}\hat{\boldsymbol{f}}(t)(dt)^{n-1}$, $\boldsymbol{\chi}_2(t) = \boldsymbol{f}_{\boldsymbol{\theta}}(t) + (a_{n-1} - b_{n-1})\int \boldsymbol{f}_{\boldsymbol{\theta}}(t)dt + \cdots + (a_1 - b_1)\int^{n-1}\boldsymbol{f}_{\boldsymbol{\theta}}(t)(dt)^{n-1}$, $\boldsymbol{\chi}_3(t) = b_{n-1}\int \boldsymbol{g}(t)dt + \cdots + b_{n-1}\int^{n-1}\boldsymbol{g}(t)(dt)^{n-1}$, and recall $\boldsymbol{\gamma}(t) = \dot{\boldsymbol{x}} - \boldsymbol{g}(t) - \boldsymbol{f}_{\boldsymbol{\theta}}(t)$. One can obtain

$$\begin{aligned}&\hat{\boldsymbol{f}}^{(n)}(t) + \boldsymbol{\chi}_1^{(n)} + a_0\hat{\boldsymbol{f}}(t)\\ =&\boldsymbol{\chi}_2^{(n)} - \boldsymbol{\chi}_3^{(n)} + a_0\boldsymbol{f}_{\boldsymbol{\theta}}(t) - b_0\boldsymbol{g}(t) - b_0\boldsymbol{f}_{\boldsymbol{\theta}}(t) + b_{n-1}\boldsymbol{x}^{(n)} + \cdots + b_0\dot{\boldsymbol{x}}.\end{aligned} \tag{C9}$$

Define an auxiliary variable $\boldsymbol{\xi} = \hat{\boldsymbol{f}}(t) + \boldsymbol{\chi}_1 - \boldsymbol{\chi}_2 + \boldsymbol{\chi}_3 - b_{n-1}\boldsymbol{x} - \cdots - b_0\int^{n-1}\boldsymbol{x}(dt)^{n-1}$. It can be rendered that

$$\begin{cases}\boldsymbol{\xi}^{(n)} = (a_0 - b_0)\boldsymbol{f}_{\boldsymbol{\theta}}(t) - a_0\boldsymbol{g}(t) - a_0\hat{\boldsymbol{f}}(t)\\ \hat{\boldsymbol{f}}(t) = \boldsymbol{\xi} - \boldsymbol{\chi}_1 + \boldsymbol{\chi}_2 - \boldsymbol{\chi}_3 + b_{n-1}\boldsymbol{x} + \cdots + b_0\int^{n-1}\boldsymbol{x}(dt)^{n-1}.\end{cases} \tag{C10}$$

Compared with Eq. (5), Eq. (C10) can capture more complex leanring residual at the cost of complex parameter adjustment.

# Section D Several special forms

In this part, we will show the feedback neural network [41], EVOLVER [22], and SEER-II [64] belong to the framework of PhyFilter. Consider the first-order form of the PhyFilter,

$$\begin{cases}\dot{\boldsymbol{\xi}} = -\boldsymbol{g}(t) - \hat{\boldsymbol{f}}(t)\\ \hat{\boldsymbol{f}}(t) = a_0\boldsymbol{\xi} + \boldsymbol{f}_{\boldsymbol{\theta}}(t) + a_0\boldsymbol{x}.\end{cases} \tag{D11}$$

## D.1 Feedback neural network

Substitute the first subequation into the second one of Eq. (D11) can yield

$$\hat{\boldsymbol{f}}(t) = -a_0\int_{t_0}^{t}(\boldsymbol{g}(t) + \hat{\boldsymbol{f}}(t))dt + \boldsymbol{f}_{\boldsymbol{\theta}}(t) + a_0\boldsymbol{x}. \tag{D12}$$

Define a state estimation $\hat{\boldsymbol{x}} = \int_{t_0}^{t}(\boldsymbol{g}(t) + \hat{\boldsymbol{f}}(t))dt$ as in [41], one can obtain

$$\hat{\boldsymbol{f}}(t) = \boldsymbol{f}_{\boldsymbol{\theta}}(t) + a_0(\boldsymbol{x} - \hat{\boldsymbol{x}}). \tag{D13}$$

The above equation is exactly the key feedback equation [41, Eq. (7)], where $\boldsymbol{f}_{\boldsymbol{\theta}}(t)$ is learned through the neural ODE by providing state trajectories and $a_0$ denotes the feedback gain.

### D.2 *Koopman* operator-based uncertainty observer

By defining $\boldsymbol{\xi}_1 = a_0\boldsymbol{\xi} + \boldsymbol{f}_{\boldsymbol{\theta}}(t)$, Eq. (D11) can be adjusted to

$$\begin{cases} \dot{\boldsymbol{\xi}}_1 = \dot{\boldsymbol{f}}_{\boldsymbol{\theta}}(t) - a_0(\boldsymbol{g}(t) + \hat{\boldsymbol{f}}(t)) \\ \hat{\boldsymbol{f}}(t) = \boldsymbol{\xi}_1 + a_0\boldsymbol{x}. \end{cases} \tag{D14}$$

By regarding $-a_0$ and $\boldsymbol{f}_{\boldsymbol{\theta}}(t)$ as the observer gain and unknown uncertainty, respectively, the above equation is exactly the uncertainty observer [22, Eq. (19)]. Note that $\boldsymbol{f}_{\boldsymbol{\theta}}(t)$ is learned through the online *Koopman* operator in [22], which involves an online construction process of the training dataset. Theoretically, the EVOLVER [22] is devised targeted at the uncertainty with $\dot{\boldsymbol{f}}(t) = \boldsymbol{h}(\boldsymbol{f}(t), \boldsymbol{x}, \boldsymbol{d})$ form, where $\boldsymbol{d}$ denotes external factors.

### D.3 *Chebyshev*-based uncertainty observer

The basic idea of the SEER-II in [64] is similar to the EVOLVER [22], apart from the term $\dot{\boldsymbol{f}}_{\boldsymbol{\theta}}(t)$ is handled by *Chebyshev* variable decomposition. The targeted uncertainty is modeled as $\boldsymbol{f}(t) = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{d})$. With the analytic assumption, it can be proven $\boldsymbol{f}(t)$ can be decomposed into $\boldsymbol{\Xi}_d\boldsymbol{\Phi}(t)\boldsymbol{\varsigma}(\boldsymbol{x})$ with arbitrarily high precision. $\boldsymbol{\Xi}_d$ is a learned parameter matrix. $\boldsymbol{\Phi}(t)$ and $\boldsymbol{\varsigma}(\boldsymbol{x})$ are composed of *Chebyshev* polynomials. Hence, $\dot{\boldsymbol{f}}_{\boldsymbol{\theta}}(t)$ can be analytically represented as

$$\dot{\boldsymbol{f}}_{\boldsymbol{\theta}}(t) = \boldsymbol{\Xi}_d\dot{\boldsymbol{\Phi}}(t)\boldsymbol{\varsigma}(\boldsymbol{x}) + \boldsymbol{\Xi}_d\boldsymbol{\Phi}(t)\frac{\partial\boldsymbol{\varsigma}(\boldsymbol{x})}{\partial\boldsymbol{x}}\dot{\boldsymbol{x}}. \tag{D15}$$

By defining $\boldsymbol{\xi}_2 = \boldsymbol{\xi}_1 - \int \boldsymbol{\Xi}_d\boldsymbol{\Phi}(t)\frac{\partial\boldsymbol{\varsigma}(\boldsymbol{x})}{\partial\boldsymbol{x}}d\boldsymbol{x}$, Eq. (D14) can be adjusted to

$$\begin{cases} \dot{\boldsymbol{\xi}}_2 = \boldsymbol{\Xi}_d\dot{\boldsymbol{\Phi}}(t)\boldsymbol{\varsigma}(\boldsymbol{x}) - a_0(\boldsymbol{g}(t) + \hat{\boldsymbol{f}}(t)) \\ \hat{\boldsymbol{f}}(t) = \boldsymbol{\xi}_2 + \int \boldsymbol{\Xi}_d\boldsymbol{\Phi}(t)\frac{\partial\boldsymbol{\varsigma}(\boldsymbol{x})}{\partial\boldsymbol{x}}d\boldsymbol{x} + a_0\boldsymbol{x} \end{cases} \tag{D16}$$

which is exactly the SEER-II [64, Eq. (11)] with observer gain $a_0$.

Compared with Eq. (D11), EVOLVER (D14) and SEER-II (D16) move differentiable learned uncertainty to the first ODE subequation. By this way, the possible discontinuous problem induced by learning can be alleviated to a certain extent.

## Section E Joint learning

A learning algorithm is presented in this part to learn the filter parameter $\boldsymbol{a}_f = \{a_0, a_1, \cdots, a_{n-1}\}$ in Eq. (3) with model parameter $\boldsymbol{\theta}$ together.

Following the definition in Section "Learning filter parameter", define $\boldsymbol{\omega} = \{\boldsymbol{a}_f, \boldsymbol{\theta}\}$. Regard the equations Eqs. (6)-(7) as an optimal control system with constrained states $\{\bar{\boldsymbol{\xi}}, \bar{\boldsymbol{\alpha}}, \bar{\boldsymbol{\beta}}\}$ and control input $\boldsymbol{\omega}$. Thus, the optimization problem to find optimal $\boldsymbol{\omega}$ can

be formalized as

$$\begin{aligned} &\min_{\boldsymbol{\omega}} J(\boldsymbol{\omega}) = \min_{\boldsymbol{\omega}} \sum_{i=1}^{N-1} l_i(\boldsymbol{f}_i^*, \hat{\boldsymbol{f}}_i, \boldsymbol{\omega}) + l_N(\boldsymbol{f}_i^*, \hat{\boldsymbol{f}}_i) \\ &s.t. \quad \boldsymbol{\varsigma}_{i+1} = \boldsymbol{\phi}_{\boldsymbol{\varsigma}}^d(\boldsymbol{\varsigma}_i, \boldsymbol{\omega}), \quad i = 1, 2, \cdots, N-1. \end{aligned} \tag{E17}$$

The *Lagrange* multiplier method is utilized to slove Eq. (E17), define *Lagrange* function

$$\mathcal{L} = J(\boldsymbol{\omega}) + \sum_{i=1}^{N-1} \boldsymbol{\lambda}_{\boldsymbol{\varsigma},i}^T (\boldsymbol{\phi}_{\boldsymbol{\varsigma},i}^d - \boldsymbol{\varsigma}_{i+1}). \tag{E18}$$

The solving procedure for optimal $\boldsymbol{\omega}$ is similar to $\boldsymbol{a}_f$ in Section "Learning filter parameter". Especially, $\nabla_{\boldsymbol{\omega}} l_i$ and $\nabla_{\boldsymbol{\omega}} \boldsymbol{\phi}_{\boldsymbol{\varsigma},i}^d$ need to be derived analytically.

# Section F Trajectory generation of aerial manipulator

Focusing on achieving the aerial interaction mission, we formulate an optimization problem to generate a 6-dimensional motion state, which includes the aerial platform's trajectory in the inertia frame and the end-effector's trajectory in its base frame. Firstly, the state propagation of the aerial manipulator is given. Let $h = \left[P_b^\top, \dot{P}_b^\top, P_{ed}^\top, \dot{P}_{ed}^\top\right]^\top \in \mathbb{R}^{12\times 1}$ be the controlled state vector. The propagation of the controlled state $h_i$ in discrete time can be formulated as

$$\begin{cases} h_{i+1} = \boldsymbol{A} h_i + \boldsymbol{B} u_i \\ y_{i+1} = \boldsymbol{C} h_{i+1} \end{cases} \tag{F19}$$

where $u_i = \left[\ddot{P}_b^\top, \ddot{P}_{ed}^\top\right]_i^\top \in \mathbb{R}^{6\times 1}$ is the optimized control action consisting of the accelerations of the aerial platform in the inertia frame and the end-effector with respect to its base. Furthermore, $y_{i+1} = \left[{y^{pu}}^\top, {y^{vu}}^\top, {y^{pm}}^\top, {y^{vm}}^\top\right]_{i+1}^\top \in \mathbb{R}^{12\times 1}$ represents the predictive model output. The tracking error of the end-effector can be formulated as

$$e_P(t+i) = P_e^d(t+i) - \hat{P}_e(t+i) \tag{F20a}$$

$$\hat{P}_e(t+i) = y^{pu}(t+i) - \boldsymbol{R}_b(t+i) P_o - \boldsymbol{R}_b(t+i) y^{pm}(t+i) \tag{F20b}$$

$$\boldsymbol{R}_b(t+i) = \boldsymbol{R}_b(t+i-1) + \delta t \boldsymbol{R}_b(t+i-1) \omega_b^\times(t) \tag{F20c}$$

where $\delta t$ is the state propagation period. $P_e^d \in \mathbb{R}^{3\times 1}$ represents the desired trajectory of the end-effector in the inertia frame. $P_o \in \mathbb{R}^{3\times 1}$ is the constant deviation between the manipulator base and the CoM of the aerial platform. $\omega_b \in \mathbb{R}^{3\times 1}$ denotes the angular velocity of the aerial platform. Therefore, the cooperative planning problem

of the aerial manipulator is converted to seek an optimal control input $u$ to minimize the following error cost function

$$Q_1 = \|e_P\|^2_{\mathbf{\Lambda_1}} = e_P^\top \mathbf{\Lambda_1} e_P, e_P \in \mathbb{R}^{3N\times 1} \tag{F21}$$

where $\mathbf{\Lambda}_1 \in \mathbb{R}^{3N\times 3N}$ is a positive-definite weighting matrix. If the optimization formulation has no constraints or does not trigger any constraints, Eq. (F21) can be simplified to an unconstrained quadratic programming (QP) problem. Nevertheless, such a pure formulation neglects practical safety during executing dynamic operations and fails to account for the motion characteristics of the aerial manipulator. Therefore, specific constraints are of seminal importance that must be incorporated in the planner design phase.

**Control action smoothness:** The smoothness of control action $u$ is a major step to prevent unnecessary aggressive control responses, thereby enhancing system safety. For such a safe concern, the following cost function for optimizing the control input $u$ is considered as

$$Q_2 = \|\Delta u\|^2_{\mathbf{\Lambda_2}} = \Delta u^\top \mathbf{\Lambda_2} \Delta u, \Delta u \in \mathbb{R}^{6N\times 1} \tag{F22}$$

where $\mathbf{\Lambda}_2 \in \mathbb{R}^{6N\times 6N}$ is a weighting matrix and $\Delta u$ represents the increment of control action at each step. The additional cost can mitigate sharp acceleration maneuvers caused by abrupt external impulses acting on the aerial manipulator.

**State stability:** Although the above two-term formulation is widely used in trajectory generation, the neglected state increments may lead to poor output stability of aerial manipulators. It is feasible to control either the aerial platform or the manipulator while maintaining end-effector tracking accuracy [30]. An undesirable scenario may arise wherein both the aerial platform and the manipulator exhibit persistent oscillatory behavior in an attempt to maintain stability of the end-effector. Therefore, a constraint on state increments is imposed to improve the stability of the optimized cooperative trajectory, which is expressed as

$$Q_{3,u} = \|\Delta y^{pu}\|^2_{\mathbf{\Lambda_3^u}} = \Delta y^{pu\top} \mathbf{\Lambda_3^u} \Delta y^{pu}, \Delta y^{pu} \in \mathbb{R}^{3N\times 1} \tag{F23a}$$

$$Q_{3,m} = \|\Delta y^{pm}\|^2_{\mathbf{\Lambda_3^m}} = \Delta y^{pm\top} \mathbf{\Lambda_3^m} \Delta y^{pm}, \Delta y^{pm} \in \mathbb{R}^{3N\times 1} \tag{F23b}$$

$$Q_3 = Q_{3,u} + Q_{3,m} \tag{F23c}$$

where $\Delta y^{pu}$ and $\Delta y^{pm}$ represent the predicted output increments of the aerial platform and the end-effector from sample time $t+1$ to $t+N$, respectively. Moreover, $\mathbf{\Lambda_3^u} \in \mathbb{R}^{3N\times 3N}$ and $\mathbf{\Lambda_3^m} \in \mathbb{R}^{3N\times 3N}$ denote the corresponding positive weighting matrices based on their motion characteristics. The inclusion of state increments significantly contributes to increasing the system safety.

The propagation matrices are formulated as

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_3 & \delta t \boldsymbol{I}_3 & \boldsymbol{0_3} & \boldsymbol{0}_3 \\ \boldsymbol{0_3} & \boldsymbol{I}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 \\ \boldsymbol{0_3} & \boldsymbol{0_3} & \boldsymbol{I}_3 & \delta t \boldsymbol{I}_3 \\ \boldsymbol{0_3} & \boldsymbol{0_3} & \boldsymbol{0}_3 & \boldsymbol{0}_3 \end{bmatrix} \boldsymbol{B} = \begin{bmatrix} 0.5\delta t^2 \boldsymbol{I}_3 & \boldsymbol{0}_3 \\ \delta t \boldsymbol{I}_3 & \boldsymbol{0}_3 \\ \boldsymbol{0_3} & 0.5\delta t^2 \boldsymbol{I}_3 \\ \boldsymbol{0_3} & \delta t \boldsymbol{I}_3 \end{bmatrix} \boldsymbol{C} = \begin{bmatrix} \boldsymbol{I}_3 & \boldsymbol{0}_3 & \boldsymbol{0_3} & \boldsymbol{0}_3 \\ \boldsymbol{0_3} & \boldsymbol{I}_3 & \boldsymbol{0}_3 & \boldsymbol{0}_3 \\ \boldsymbol{0_3} & \boldsymbol{0_3} & \boldsymbol{I}_3 & \boldsymbol{0}_3 \\ \boldsymbol{0_3} & \boldsymbol{0_3} & \boldsymbol{0}_3 & \boldsymbol{I}_3 \end{bmatrix} \tag{F24}$$

where $\delta t = 0.07$. Moreover, the optimization parameters of the presented cooperative planner are: $\mathbf{\Lambda_1} = \text{diag}\{6000, 6000, 9000\}$, $\mathbf{\Lambda_2} = \text{diag}\{700, 700, 700, 1, 1, 1\}$, $\mathbf{\Lambda_3^u} = \text{diag}\{20, 20, 20\}$, $\mathbf{\Lambda_3^m} = \text{diag}\{10, 10, 10\}$.

# Section G Experimental hardware

In this section, the hardware components utilized in experiments, including a quadruped robot, an aerial drone, and an aerial manipulator, are introduced.

For the locomotion experiment, the DEEPRobotics Lite3 quadruped robot is employed, which comprises a locomotion host (equipped with an RK3588 CPU) and a perception host (equipped with an NVIDIA Jetson Xavier NX). The robot has a weight of 11 kg and dimensions of $548 \times 370 \times 118$ mm$^3$. During employment, the locomotion control policy is executed on a remote personal computer, with control commands transmitted to the robot via a Wi-Fi connection.

In the experiments involving drone flight and aerial manipulation, the same coaxial dual-rotor octocopter platform is utilized. For state perception, an STM32F427 microprocessor is employed to receive data from both the motion capture system and the onboard IMU, while another STM32F427 microprocessor is dedicated to low-level control. A UWB (Ultra-Wideband) communication module is employed for signal transmission between the drone and the ground station. To enable online planning of pick-and-place trajectories, an Intel NUC is integrated as the onboard computing unit for the aerial manipulator. Additionally, a 5-DOF manipulator is mounted on the drone to execute aerial manipulation tasks.

# References


[1] Wensing, P.M., Wang, A., Seok, S., Otten, D., Lang, J., Kim, S.: Proprioceptive actuator design in the mit cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots. IEEE Transactions on Robotics **33**(3), 509–522 (2017)

[2] Todorov, E., Erez, T., Tassa, Y.: Mujoco: A physics engine for model-based control. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5026–5033 (2012)

[3] Makoviychuk, V., Wawrzyniak, L., Guo, Y.R., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., State, G.: Isaac gym: high performance gpu-based physics simulation for robot learning. arXiv preprint arXiv:2108.10470 (2021)

[4] Ha, S., Lee, J., Panne, M., Xie, Z., Yu, W., Khadiv, M.: Learning-based legged locomotion: State of the art and future perspectives. The International Journal of Robotics Research **44**(8), 1396–1427 (2025)

[5] Han, L., Zhu, Q., Sheng, J., Zhang, C., Li, T., Zhang, Y., Zhang, H., Liu, Y., Zhou, C., Zhao, R., *et al.*: Lifelike agility and play in quadrupedal robots using reinforcement learning and generative pre-trained models. Nature Machine Intelligence **6**(7), 787–798 (2024)

[6] Yang, C., Yuan, K., Zhu, Q., Yu, W., Li, Z.: Multi-expert learning of adaptive legged locomotion. Science Robotics **5**(49), 2174 (2020)

[7] Kumar, A., Fu, Z., Pathak, D., Malik, J.: RMA: Rapid motor adaptation for legged robots. In: Proceedings of the Robotics: Science and Systems (2021)

[8] Lee, J., Bjelonic, M., Reske, A., Wellhausen, L., Miki, T., Hutter, M.: Learning robust autonomous navigation and locomotion for wheeled-legged robots. Science Robotics **9**(89), 9641 (2024)

[9] Kim, H., Oh, H., Park, J., Kim, Y., Youm, D., Jung, M., Lee, M., Hwangbo, J.: High-speed control and navigation for quadrupedal robots on complex and discrete terrain. Science Robotics **10**(102), 6192 (2025)

[10] Shi, F., Zhang, C., Miki, T., Lee, J., Hutter, M., Coros, S.: Rethinking robustness assessment: Adversarial attacks on learning-based quadrupedal locomotion controllers. In: Proceedings of Robotics: Science and Systems (2024)

[11] Wensing, P.M., Posa, M., Hu, Y., Escande, A., Mansard, N., Prete, A.D.: Optimization-based control for dynamic legged robots. IEEE Transactions on Robotics **40**, 43–63 (2023)

[12] Lyu, S., Lang, X., Zhao, H., Zhang, H., Ding, P., Wang, D.: RL2AC: Reinforcement learning-based rapid online adaptive control for legged robot robust locomotion. In: Proceedings of the Robotics: Science and Systems (2024)

[13] Humphreys, J., Zhou, C.: Learning to adapt: Bio-inspired gait strategies for versatile quadruped locomotion. arXiv preprint arXiv:2412.09440 (2024)

[14] Jenelten, F., He, J., Farshidian, F., Hutter, M.: Dtc: Deep tracking control. Science Robotics **9**(86), 5401 (2024)

[15] Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., Scaramuzza, D.: Champion-level drone racing using deep reinforcement learning. Nature **620**(7976), 982–987 (2023)

[16] Tobin, J., Biewald, L., Duan, R., Andrychowicz, M., Handa, A., Kumar, V., McGrew, B., Ray, A., Schneider, J., Welinder, P., *et al.*: Domain randomization and generative models for robotic grasping. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3482–3489 (2018)

[17] Ferede, R., De Wagter, C., Izzo, D., Croon, G.C.H.E.: End-to-end reinforcement learning for time-optimal quadcopter flight. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 6172–6177 (2024)

[18] Song, Y., Romero, A., Müller, M., Koltun, V., Scaramuzza, D.: Reaching the limit in autonomous racing: Optimal control versus reinforcement learning. Science Robotics **8**(82), 1462 (2023)

[19] Cao, H., Shen, J., Zhang, Y., Fu, Z., Liu, C., Sun, S., Zhao, S.: Proximal cooperative aerial manipulation with vertically stacked drones. Nature, 1–8 (2025)

[20] Saviolo, A., Loianno, G.: Learning quadrotor dynamics for precise, safe, and agile flight control. Annual Reviews in Control **55**, 45–60 (2023)

[21] Huang, K., Rana, R., Spitzer, A., Shi, G., Boots, B.: DATT: Deep adaptive trajectory tracking for quadrotor control. In: Proceedings of Conference on Robot Learning (2023)

[22] Jia, J., Zhang, W., Guo, K., Wang, J., Yu, X., Shi, Y., Guo, L.: EVOLVER: Online learning and prediction of disturbances for robot control. IEEE Transactions on Robotics **40**, 382–402 (2023)

[23] Zhang, W., Liu, Q., Wang, M., Jia, J., *et al.*: Design of an aerial manipulator system applied to capture missions. In: Proceedings of International Conference on Unmanned Aircraft Systems, pp. 1063–1069 (2021)

[24] Ollero, A., Tognon, M., Suarez, A., Lee, D., Franchi, A.: Past, present, and future of aerial robotic manipulators. IEEE Transactions on Robotics **38**(1), 626–645 (2022)

[25] Aucone, E., Mintchev, S.: Embodied aerial physical interaction: Combining body and brain for robust interaction with unstructured environments. Science Robotics **10**(102), 0200 (2025)

[26] Huang, Y., Ke, J., Zhang, X., Ota, J.: Dynamic parameter identification of serial robots using a hybrid approach. IEEE Transactions on Robotics **39**(2), 1607–1621 (2023)

[27] Wang, M., Lyu, S., Liu, Q., Yang, Z., Guo, K., Yu, X.: Precise end-effector control for an aerial manipulator under composite disturbances: Theory and experiments. IEEE Transactions on Automation Science and Engineering **22**, 4006–4021 (2024)

[28] He, G., Guo, X., Tang, L., Zhang, Y., Mousaei, M., Xu, J., Geng, J., Scherer, S., Shi, G.: Flying hand: End-effector-centric framework for versatile aerial manipulation teleoperation and policy learning. In: Proceedings of Robotics: Science and Systems, LosAngeles, CA, USA (2025)

[29] Lee, D., Seo, H., Jang, I., Lee, S.J., Kim, H.J.: Aerial manipulator pushing a movable structure using a DOB-based robust controller. IEEE Robotics and Automation Letters **6**(2), 723–730 (2020)

[30] Wang, M., Chen, Z., Guo, K., Yu, X., Zhang, Y., Guo, L., Wang, W.: Millimeter-level pick and peg-in-hole task achieved by aerial manipulator. IEEE Transactions on Robotics **40**, 1242–1260 (2023)

[31] Zhang, D., Loquercio, A., Tang, J., Wang, T.-H., Malik, J., Mueller, M.W.: A learning-based quadcopter controller with extreme adaptation. IEEE Transactions on Robotics **41**, 3948–3964 (2025)

[32] Wu, Y., Zhou, Z., Wei, M., Xie, L., Liu, R., Cheng, H.: Learning variable whole-body control for agile aerial manipulation in strong winds. IEEE Robotics and Automation Letters **10**(5), 4794–4801 (2025)

[33] Wang, K., Lai, G., Yu, Y., Du, J., Sun, J., Xu, B., Franchi, A., Sun, F.: Versatile tasks on integrated aerial platforms using only onboard sensors: Control, estimation, and validation. IEEE Transactions on Robotics **41**, 3518–3538 (2025)

[34] Chen, Y., Liang, J., Wu, Y., Miao, Z., Zhang, H., Wang, Y.: Adaptive sliding-mode disturbance observer-based finite-time control for unmanned aerial manipulator with prescribed performance. IEEE Transactions on Cybernetics **53**(5), 3263–3276 (2023)

[35] Jia, J., Wang, M., Liu, Y., Guo, K., Yu, X., Xie, L., Guo, L.: Disturbance observer for estimating coupled disturbances. arXiv preprint arXiv:2407.13229 (2024)

[36] Byun, J., Jang, I., Lee, D., Kim, H.J.: A hybrid controller enhancing transient performance for an aerial manipulator extracting a wedged object. IEEE Transactions on Automation Science and Engineering **21**(3), 3264–3273 (2024)

[37] Xiu, B., Li, Z., Pang, B., Liu, H., Yu, X.: Integral anti-disturbance control for unmanned aerial manipulator based on the characteristic model. IEEE Transactions on Circuits and Systems I: Regular Papers (2025)

[38] Cao, H., Li, Y., Liu, C., Zhao, S.: ESO-based robust and high-precision tracking control for aerial manipulation. IEEE Transactions on Automation Science and Engineering **21**(2), 2139–2155 (2024)

[39] Cao, H., Wu, Y., Wang, L.: Adaptive NN motion control and predictive coordinate planning for aerial manipulators. Aerospace Science and Technology **126**, 107607 (2022)

[40] Cai, K., Yu, H., Zhang, Z., Liang, X., Fang, Y., Han, J.: An experiment study for unmanned aerial manipulator systems with L1 adaptive augmentation of geometric control. Control Engineering Practice **164**, 106418 (2025)

[41] Jia, J., Yang, Z., Wang, M., Guo, K., Yang, J., Yu, X., Guo, L.: Feedback favors the generalization of neural ODEs. In: Proceedings of International Conference on Learning Representations (2025)

[42] O'Connell, M., Shi, G., Shi, X., Azizzadenesheli, K., Anandkumar, A., Yue, Y., Chung, S.-J.: Neural-fly enables rapid learning for agile flight in strong winds. Science Robotics **7**(66), 6597 (2022)

[43] Lupu, E.S., Xie, F., Preiss, J.A., Alindogan, J., Anderson, M., Chung, S.-J.: Magicvfm-meta-learning adaptation for ground interaction control with visual foundation models. IEEE Transactions on Robotics **41**, 180–199 (2024)

[44] Han, J.: From PID to active disturbance rejection control. IEEE Transactions on Industrial Electronics **56**(3), 900–906 (2009)

[45] Seeber, R., Haimovich, H., Horn, M., Fridman, L.M., De Battista, H.: Robust exact differentiators with predefined convergence time. Automatica **134**, 109858 (2021)

[46] Yu, H., Guo, D., Yin, H., Chen, A., Xu, K., Chen, Z., Wang, M., Tan, Q., Wang, Y., Xiong, R.: Neural motion prediction for in-flight uneven object catching. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4662–4669 (2021)

[47] Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., Yang, L.: Physics-informed machine learning. Nature Reviews Physics **3**(6), 422–440 (2021)

[48] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P.: Domain randomization for transferring deep neural networks from simulation to the real world. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 23–30 (2017)

[49] Chen, R.T., Rubanova, Y., Bettencourt, J., Duvenaud, D.K.: Neural ordinary differential equations. Advances in Neural Information Processing Systems **31** (2018)

[50] Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics **378**, 686–707 (2019)

[51] Wang, C., Ji, K., Geng, J., Ren, Z., Fu, T., Yang, F., Guo, Y., He, H., Chen, X., Zhan, Z., Du, Q., Su, S., Li, B., Qiu, Y., Du, Y., Li, Q., Yang, Y., Lin, X., Zhao, Z.: Imperative learning: A self-supervised neural-symbolic learning framework for robot autonomy. The International Journal of Robotics Research, 02783649251353181 (2024)

[52] Greydanus, S., Dzamba, M., Yosinski, J.: Hamiltonian neural networks. Advances in Neural Information Processing Systems **32** (2019)

[53] Cranmer, M., Greydanus, S., Hoyer, S., Battaglia, P., Spergel, D., Ho, S.: Lagrangian neural networks. arXiv preprint arXiv:2003.04630 (2020)

[54] Lutter, M., Peters, J.: Combining physics and deep learning to learn continuous-time dynamics models. The International Journal of Robotics Research **42**(3), 83–107 (2023)

[55] Romero, A., Song, Y., Scaramuzza, D.: Actor-critic model predictive control. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 14777–14784 (2024)

[56] Li, W., Zhao, H., Yu, Z., Du, Y., Zou, Q., Hu, R., Xu, K.: PIN-WM: Learning physics-informed world models for non-prehensile manipulation. Proceedings of the Robotics: Science and Systems (2025)

[57] Geng, H., Wang, F., Wei, S., Li, Y., Wang, B., An, B., Cheng, C.T., Lou, H., Li, P., Wang, Y.-J., Liang, Y., Goetting, D., Xu, C., Chen, H., Qian, Y., Geng, Y., Mao, J., Wan, W., Zhang, M., Lyu, J., Zhao, S., Zhang, J., Zhang, J., Zhao, C., Lu, H., Ding, Y., Gong, R., Wang, Y., Kuang, Y., Wu, R., Jia, B., Sferrazza, C., Dong, H., Huang, S., Wang, Y., Malik, J., Abbeel, P.: Roboverse: Towards a unified platform, dataset and benchmark for scalable and generalizable robot

learning. In: Proceedings of Robotics: Science and Systems (2025)

[58] Green, M., Limebeer, D.J.: Linear Robust Control. Courier Corporation, Chelsea, MA (2012)

[59] Wu, K., Hou, C., Liu, J., Che, Z., Ju, X., Yang, Z., Li, M., Zhao, Y., Xu, Z., Yang, G., Zhao, Z., Li, G., Jin, Z., Wang, L., Mao, J., Wang, X., Fan, S., Liu, N., Ren, P., Zhang, Q., Lyu, Y., Liu, M., He, J., Luo, Y., Gao, Z., Li, C., Gu, C., Fu, Y., Wu, D., Wang, X., Chen, S., Wang, Z., An, P., Qian, S., Zhang, S., Tang, J.: Robomind: Benchmark on multi-embodiment intelligence normative data for robot manipulation. arXiv preprint arXiv:2412.13877 (2024)

[60] Kouw, W.M., Loog, M.: A review of domain adaptation without target labels. IEEE Transactions on Pattern Analysis and Machine Intelligence **43**(3), 766–785 (2019)

[61] Slotine, J.-J.E., Li, W., *et al.*: Applied Nonlinear Control vol. 199. Prentice Hall, Englewood Cliffs, NJ (1991)

[62] Chidlovskii, B., Clinchant, S., Csurka, G.: Domain adaptation in the absence of source domain data. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 451–460 (2016)

[63] Yin, H., Chen, R., Wang, Y., Xiong, R.: Rall: End-to-end radar localization on lidar map using differentiable measurement model. IEEE Transactions on Intelligent Transportation Systems **23**(7), 6737–6750 (2021)

[64] Jia, J., Guo, K., Wang, Y., Zhou, S., Zhang, J., Liu, Y., Yu, X., Shi, Y., Guo, L.: FORESEER: Recognize and utilize uncertainties by integrating data-based learning and symbolic feedback. The International Journal of Robotics Research, 02783649251364000 (2025)